==============================

Generating Multi-Sentence Abstractive Summaries of Interleaved Texts

Anonymous authors
Paper under double-blind review

Abstract

In multi-participant postings, as in online chat conversations, several conversations or topic threads may take place concurrently. This leads to difficulties for readers reviewing the postings in not only following discussions but also in quickly identifying their essence. A two-step process, disentanglement of interleaved posts followed by summarization of each thread, addresses the issue, but disentanglement errors are propagated to the summarization step, degrading the overall performance. To address this, we propose an end-to-end trainable encoder-decoder network for summarizing interleaved posts. The interleaved posts are encoded hierarchically, i.e., word-to-word (words in a post) followed by post-to-post (posts in a channel). The decoder also generates summaries hierarchically, thread-to-thread (generate thread representations) followed by word-to-word (i.e., generate summary words). Additionally, we propose a hierarchical attention mechanism for interleaved text. Overall, our end-to-end trainable hierarchical framework enhances performance over a sequence to sequence framework by 8-10% on multiple synthetic interleaved texts datasets.

Introduction

Interleaved texts are becoming more common with new ways of working and new forms of communication, e.g., multi-author entries for activity reports, meeting minutes and social media conversations, such as Slack. Quickly getting a sense of or following the content of different threads in interleaved texts, where posts belonging to different threads occur in one sequence, is often difficult. An example of two threads with multiple posts interleaved to form a sequence is:

Post1-Thread1 $\rightarrow$ Post1-Thread2 $\rightarrow$ Post2-Thread1 $\rightarrow$ Post3-Thread1 $\rightarrow$ Post2-Thread2 $\rightarrow$ Post3-Thread2

This intermingling leads to difficulties in not only following discussions but also in retrieving their essence. In conversation disentanglement, interleaved posts are grouped by the thread; e.g., the previous example could be rearranged as:

Post1-Thread1 $\rightarrow$ Post2-Thread1 $\rightarrow$ Post3-Thread1
Post1-Thread2 $\rightarrow$ Post2-Thread2 $\rightarrow$ Post3-Thread2

In analyzing interleaved texts, Shang et al. (2018) went a step further and proposed summarization of the interleaved texts. They designed an unsupervised two-step system and evaluated the system on meeting texts. In the first step, a conversation disentanglement component disentangles the texts thread-wise. Then, in the second step, a multi-sentence compression component compresses the thread-wise posts to single sentence summaries. However, this system has a major disadvantage, in that the disentanglement obtained through either supervised Jiang et al. (2018) or unsupervised Wang & Oard (2009) methods propagate its errors to the downstream summarization task, and thus, degrades the overall performance.

We aim to tackle this issue of error propagation through an end-to-end trainable encoder-decoder system that takes a variable length input, e.g., interleaved texts, processes it and generates a variable length output, e.g., a multi-sentence summary; see Figure 1 for an illustration. An end-to-end system eliminates the disentanglement component, and thus, the error propagation.

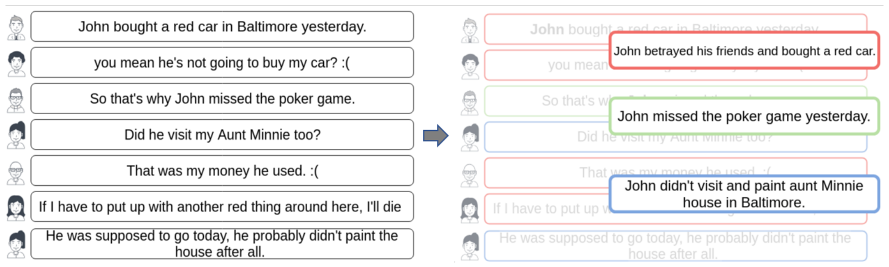
John bought a red car in Baltimore yesterday.
you mean he's not going to buy my car? :(
So that's why John missed the poker game.
Did he visit my Aunt Minnie too?
That was my money he used. :(
If I have to put up with another red thing around here, I'll die
He was supposed to go today, he probably didn't paint the house after all.
John betrayed his friends and bought a red car.
John missed the poker game yesterday.
John didn't visit and paint aunt Minnie house in Baltimore.

Figure 1: 7 interleaved posts are implicitly disentangled into 3 threads, and single sentence summaries are generated for each thread. Posts are outlined with colors corresponding the color of each summary.

The encoder first performs word-to-word encoding to embed each post, followed by post-to-post encoding to embed the overall content of the posts and represent the discourse structure of the interleaved texts. The decoder has a thread-to-thread decoder to generate a representation for each thread, and the thread representation is given to a word-to-word decoder to generate a summary sentence. We also propose to use hierarchical attention similar to Nallapati et al. (2017), but instead of computing post-level attention at every word, attentions are only computed when decoding new sentences. Further, the attention networks are trained end-to-end.

Despite the availability of a multitude of real-world interleaved texts, a major challenge to train encoder-decoder models is the lack of labels (summaries). As labeling is difficult and expensive Barker et al. (2016); Aker et al. (2016); Verberne et al. (2018), we synthesize two separate corpora. Each is derived by mixing texts and associated summaries from a corpus of documents, where the mixed text has a structure reflective of a multi-party conversation with inter-leaved threads and the summary highlights the information in the threads. We find abstracts and titles of randomized controlled trial (RCT) articles, a PubMed corpus, and also for Stack Exchange posts and titles.

To summarize, our contributions are threefold:

- We propose to combine a hierarchical encoder and decoder to obtain multi-sentence summaries of interleaved texts.
- We propose a novel hierarchical attention mechanism that is integrated with the hierarchical encoder-decoder architecture and which equips the decoder to disentangle the interleaving further.
- We use two synthetic datasets to verify the ideas and show our end-to-end trainable architecture addresses not only the issue of error-propagation observed in competitive methods but also improves the performance.

Related Work

Quite often multi-party conversations, e.g. news comments, social media conversation and activity report, have tens of posts discussing several different matters pertaining to a subject. Ma et al. (2012); Aker et al. (2016); Shang et al. (2018) designed methodologies to summarize posts in order to provide an overview on the discussed matters. They broadly follow the same approach: cluster the posts and then extract a summary from each cluster.

There are two kinds of summarization: abstractive and extractive. In abstractive summarization, the model utilizes a corpus level vocabulary and generates novel sentences as the summary, while extractive models extract or rearrange the source words as the summary. Abstractive models based on neural sequence-to-sequence (seq2seq) Rush et al. (2015) proved to generate summaries with higher ROUGE scores than the feature-based abstractive models. Integration of attention into seq2seq Bahdanau et al. (2014) led to further advancement of abstractive summarization Nallapati et al. (2016); Chopra et al. (2016).

There are many possible patterns of organization of the information in texts, e.g., chronological pattern. News articles have an inverted pyramid pattern, i.e., core information in the lead sentences and the extra information in later sentences. A seq2seq model is appropriate for summarization of a news article as it encodes and decodes sequentially. However, in interleaved texts, related information maybe separated; thus a seq2seq model may be competent but not sufficient.

Li et al. (2015) proposed an encoder-decoder (auto-encoder) model that utilizes a hierarchy of networks: word-to-word followed by sentence-to-sentence. Their model is better at capturing the underlying structure than a vanilla sequential encoder-decoder model (seq2seq). Krause et al. (2017); Jing et al. (2018) showed multi-sentence captioning of an image through a decoder based on a hierarchical Recurrent Neural Network (RNN), topic-to-topic followed by word-to-word, is better than seq2seq.

These works suggest a hierarchical encoder, with word-to-word encoding followed by post-to-post, will better recognize the dispersed information in interleaved texts. Similarly, a hierarchical decoder, thread-to-thread followed by word-to-word, will intrinsically disentangle the posts, and therefore, generate more appropriate summaries.

Nallapati et al. (2016) devised a hierarchical attention mechanism for a seq2seq model, where two levels of attention distributions over the source, i.e., sentence and word, are computed at every step of the word decoding. Based on the sentence attentions, the word attentions are rescaled. Hsu et al. (2018) slightly simplified this mechanism and computed the sentence attention only at the first step. Our hierarchical attention is more intuitive and computes new sentence attentions for every new summary sentence, and unlike Hsu et al. (2018), is trained end-to-end .

Model

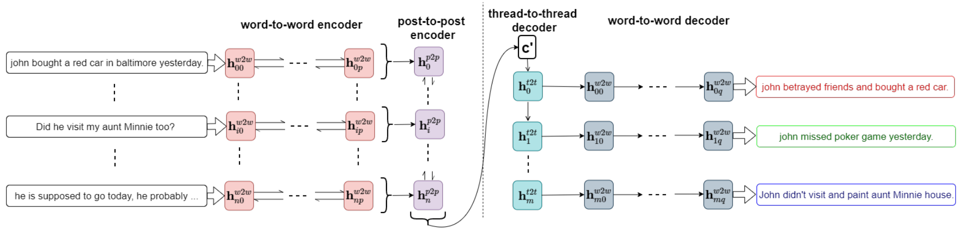

Figure 2: Hierarchical encoder-decoder architecture.

Problem Statement

We aim to design a system that when given a sequence of posts, $C = \langle P_1, \ldots, P_{|C|} \rangle$, produces a sequence of summaries, $T = \langle S_1, \ldots, S_{|T|} \rangle$. For simplicity and clarity, unless otherwise noted, we will use lowercase italics for variables, uppercase italics for sequences, lowercase bold for vectors and uppercase bold for matrices.

Figure 2 illustrates the proposed hierarchical encoder-decoder framework. In the framework, first, a low-level, word-to-word encoder converts a sequence of words in a post, P_j, to a sequence of representations, $H_j = \langle \mathbf{h}_{j0}^{w2w}, \ldots, \mathbf{h}_{j|P_j|}^{w2w} \rangle$. Subsequently, a top-level, post-to-post encoder converts those representations, $\langle H_0, \ldots, H_{|C|} \rangle$, to a sequence of top-level post representations $\langle \mathbf{h}_1^{p2p}, \ldots, \mathbf{h}_{|C|}^{p2p} \rangle$. These encoded representations are then passed to the decoder, which utilizes a top-level, thread-to-thread, decoder to disentangle them into a sequence of thread representations $\langle \mathbf{h}_1^{t2t}, \ldots, \mathbf{h}_{|T|}^{t2t} \rangle$. Finally, a low-level, word-to-word, decoder takes a feed-forward mapped thread representation $\mathbf{h}_i^{t2t}$ and generates a sequence of summary words $\langle w_{i1}, \ldots, w_{i|S_i|} \rangle$.

The maximum number of posts in a sequence of interleaved texts is denoted by n and threads by m. We limit the number of words in posts and summaries to fixed lengths by either truncating or padding, and denote them by p and q respectively. The hidden states of the encoder and decoder have dimensionality l.

ENCODER

The hierarchical encoder used in Figure 2 is based on Nallapati et al. (2017), where word-to-word and post-to-post encoders are bi-directional LSTMs. The word-to-word Bi-LSTM encoder (E_{w2w}) runs over word embeddings of any post, P_j, and generates a set of hidden representations ($\langle \mathbf{h}_{j0}^{E_{w2w}}, \dots, \mathbf{h}_{jp}^{E_{w2w}} \rangle$); see left middle section in Figure 2. The average pooled value of the word-to-word representations of post P_j ($\frac{1}{p}\sum_{k=0}^{p} \mathbf{h}_{jk}^{E_{w2w}}$) are input to the post-to-post Bi-LSTM encoder (E_{t2t}), which then generates a set of representations ($\langle \mathbf{h}_0^{E_{p2p}}, \dots, \mathbf{h}_n^{E_{p2p}} \rangle$) corresponding to those posts. In summary, the encoding process results into a tensor of word-to-word representations, $\mathbf{W}$, with dimension $n \times p \times 2l$ and a matrix of post-to-post representations, $\mathbf{P}$, with dimension $n \times 2l$ for a given channel, C.

DECODER

Our hierarchical decoder (see Figure 2) is based on Li et al. (2015), where both thread-to-thread and word-to-word decoders are uni-directional LSTMs.

The initial state $\mathbf{h}_0^{D_{t2t}}$ of the thread-to-thread LSTM decoder ($f^{D_{t2t}}$) is set with a feedforward-mapped representation of an average pooled post representations ($\mathbf{c}' = \frac{1}{n}\sum_{k=0}^{n} \mathbf{h}_k^{p2p}$). At each step i of the decoder, a sequence of attention weights, $\langle \hat{\beta}_{i,0}, \dots, \hat{\beta}_{i,n\times p} \rangle$, corresponding to the set of encoded word representations, $\langle \mathbf{h}_0^{w2w}, \dots, \mathbf{h}_{n\times p}^{w2w} \rangle$ are computed utilizing the previous state, $\mathbf{h}_{i-1}^{D_{t2t}}$. We will elaborate the attention computation in the next section.

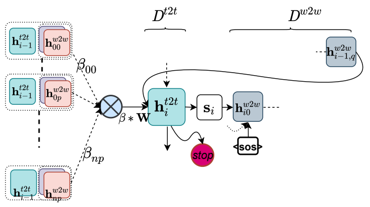

Figure 3: A step in the thread-to-thread decoder.

A weighted representation of the words (crossed blue circle) is then computed: $\sum_{j=1}^{n} \hat{\beta_{ij}} \mathbf{W}_{ij}$, and used as one input to LSTM $f^{D_{t2t}}$. Additionally, we use the last hidden state $\mathbf{h}_{i-1.}^{D_{w2w}}$ of the word-to-word decoder LSTM (D_{w2w}) of the previously generated summary sentence as the second input to compute the next state of thread-to-thread decoder, i.e., $\mathbf{h}_i^{D_{t2t}}$. The motivation is to provide information about the previous sentence.

The current state $\mathbf{h}_i^{D_{t2t}}$ is passed through a single layer feedforward network and a distribution over STOP=1 and CONTINUE=0 is computed:

$$p_i^{STOP} = \sigma(\mathrm{g}\left(\mathbf{h}_i^{D_{t2t}}\right)) \tag{1}$$

where g is a feedforward network. In Figure 3, the process is depicted by a red circle. The thread-to-thread decoder keeps decoding until p_i^{STOP} is greater than 0.5.

Additionally, the current state $\mathbf{h}_i^{D_{t2t}}$ is passed through another single layer feedforward network k followed by $\tanh$ activation to compute the thread representation $\mathbf{s}_i = \tanh(\mathrm{k}\left(\mathbf{h}_i^{D_{t2t}}\right))$.

Given a thread representation $\mathbf{s}_i$, the word-to-word decoder generates a summary for the thread. Our word-to-word decoder is based on Bahdanau et al. (2014). It is a unidirectional attentional LSTM ($f^{D_{w2w}}$); see the right-hand side of Figure 2. We refer to Bahdanau et al. (2014) for further details.

HIERARCHICAL ATTENTION

Our novel dynamic hierarchical attention is based on the idea of making attention distributions interdependent. We draw inspiration from See et al. (2017) wherein a model learns to merge attention distribution over source tokens with the attention distribution over vocabulary tokens using switch probabilities, and thereby, determines which tokens to copy and which to pull from vocabulary in the process of generating a summary.

In our hierarchical attention, the higher level attention (corresponding to source tokens), $\boldsymbol{\beta}$, that is computed while obtaining a thread representation, $\mathbf{s}$, scales the lower level attention (also corresponding to source tokens), $\boldsymbol{\alpha}$, that is computed while generating a word, y, of a summary, S; see yellow rectangles in Figure 4.

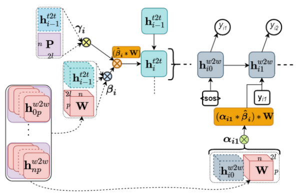

Figure 4: Hierarchical attention mechanism for a hierarchical encoder.

At step i of thread-to-thread decoding, we compute elements of $\boldsymbol{\beta}_i$ as below.

$$\beta_{ik} = \sigma(\text{attn}^{\beta}(\mathbf{h}_{i-1}^{D_{t2t}}, \mathbf{a}_k)) \quad k \in \{1, \dots, n \times p\} \tag{2}$$

$$\begin{aligned} \mathbf{a}_k &= \text{add}(\mathbf{W}_{jl}, \mathbf{P}_j) \\ k &:= j * l, \text{where } \; j \in \{1, \dots, n\}, \;\; l \in \{1, \dots, p\} \end{aligned} \tag{3}$$

where $\mathbf{a}_k$ is computed as in Eq. 3. $\mathbf{W}$ in Eq. 3 is a word-to-word encoder representations of dimension $n \times p \times 2l$, add aligns a post representation to its constituting word representations and does element-wise addition, and $attn^{\beta}$ is a feedforward network that aligns the current thread decoder state $\mathbf{h}_{i-1}^{D_{t2t}}$ with all $n * p$ number of $\mathbf{a}_k$ vectors.

Further, at step i of thread-to-thread decoding, we also compute elements of post-level attention, i.e., $\boldsymbol{\gamma}_i$ as below.

$$\gamma_{iq} = \sigma(\text{attn}^{\beta}(\mathbf{h}_{i-1}^{D_{t2t}}, \mathbf{P}_q) \quad q \in \{1, \dots, n\} \tag{4}$$

Then, we use $\boldsymbol{\gamma}$ to rescale high-level attention, $\boldsymbol{\beta}$ as below.

$$\begin{aligned} \hat{\beta}_{ik} &= \beta_{ik} * \gamma_{il} \\ k &:= j * l, \text{where } \; j \in \{1, \dots, p\} \;\; \& l \in \{1, \dots, n\} \end{aligned} \tag{5}$$

At step t of word-to-word decoding, we compute elements of $\boldsymbol{\alpha}_t$ as below.

$$\alpha_{tk} = \frac{\exp(\mathbf{e}_{tk})}{\sum_{k=1}^{n \cdot p} \exp(\mathbf{e}_{tk})} \tag{6}$$

$$\mathbf{e}_{jk} = \text{attn}^{\alpha}(\mathbf{h}_{t-1}^{D_{w2w}}, \mathbf{a}_k) \tag{7}$$

where $\mathbf{a}_k$ is same as in Eq. 3 and $attn^{\alpha}$ is a feedforward network that aligns the current word decoder state $\mathbf{h}_{t-1}^{D_{w2w}}$ with all $n * p$ number of $\mathbf{a}_k$ vectors.

Finally, we use rescaled high-level word attentions, $\hat{\boldsymbol{\beta}}$, for rescaling a low level attention:

$$\hat{\alpha}_{jk} = \frac{\hat{\beta}_k \times \exp(\mathbf{e}_{jk})}{\sum_{k=1}^{n \cdot p} \hat{\beta}_k \times \exp(\mathbf{e}_{jk})} \tag{8}$$

Table 1 compares different forms of hierarchical attention.

Model	End2End	HierDcode(many β)	reuse
Nallapati et al.	yes	no	no
Hsu et al.	no	no	no
Li, Luong, and Jurafsky	yes	yes	no
Ours	yes	yes	yes

Table 1: Comparing different forms of hierarchical attention. HierDcode=Hierarchical Decoding

Training Objective

We train our hierarchical encoder-decoder network similarly to an attentive seq2seq model Bahdanau et al. (2014), but with an additional weighted sum of sigmoid cross-entropy loss on stopping distribution; see Eq. 1. Given a summary, $Y_i = \langle w_{i0}, \ldots, w_{iq} \rangle$, our word-to-word decoder generates a target $\hat{Y} = \langle y_{i0}, \ldots, y_{iq} \rangle$, with words from a same vocabulary U. We train our model end-to-end by minimizing the objective given in Eq. 9.

$$\sum_{i=1}^{m}\sum_{j=1}^{q} \log p_\theta \left(y_{ij} | w_{i \cdot <j}, \mathbf{W} \right) + \lambda \sum_{i=1}^{m} y_i^{STOP} \log(p_i^{STOP}) \tag{9}$$

Dataset

Obtaining labeled training data for conversation summarization is challenging. The available ones are either extractive (Verberne et al. (2018)) or too small (Barker et al. (2016); Anguera et al. (2012)) to train a neural model. To get around this issue and verify the proposed architecture, we synthesized a dataset by utilizing a corpus of conventional texts for which summaries are available. The PubMed corpus contains a type of article, i.e., randomized controlled trials (RCT), where sentences in the abstract are structured into sections and the title of the article is an abstractive summary of the information from these sections (Dernoncourt & Lee (2017)). Further, Stack Exchange is a question-and-answer site for several diverse fields topics. In this dataset, every question (with multiple sentences) has a title, and zero, one or many answers. The title forms the gist or abstract of the question. Thus, a random interleaving of sentences from a few pubmed abstracts or Stack Exchange questions roughly resembles interleaved texts, and correspondingly interleaving of titles resembles its multi-sentence summary.

The algorithm that we devised for creating synthetic interleaved texts is defined in detail in the Appendix. Given a set of abstracts and titles and a range for , a set of concatentated interleaved texts and summaries are returned. The number of abstracts to include in the interleaved texts is given as a range (from a to b) and the number of sentences per abstract to include is given as a second range (from m to n).

We vary Interleave parameters as below, and create three different corpora for experiments:

- Easy: a=2, b=2, m=5 and n=5
- Medium: a=2, b=3, m=2 and n=5
- Hard: a=2, b=5, m=2 and n=5

Table 2 shows an example of a data instance in the Hard Interleaved RCT corpus.

Experiments

Evaluation Metrics: we report ROUGE-1 (unigram), ROUGE-2 (bigram), and ROUGE-L (longest-common substring) as the quantitative evaluation of the models.

Parameters: Pubmed Medium and Hard corpora train, test and validation have approximately 290k, 6k and 1.5k instances, respectively. In case of Stack-Exchange, the medium corpus train has 180k and Hard has 210k instances, while the both corpora have test and validation of approximately 5k and 4k instances, respectively. Remaining hyper-parameters is described in detail in the Appendix.

A_2	this study was conducted to evaluate the influence of e…
A_1	to assess the effect of a program of supervised fitness…
A_1	an 8-week randomized , controlled trial ..…
A_2	nine endurance-trained athletes participated in a randomised…
⋮	⋮
A_0	we examined the effects of intensity of training on ratings…
A_0	subjects were recruited as sedentary controls or were randomly…
A_0	the at lt group trained at velocity lt and the greater than…
A_1	data were obtained on 47 of 51 intervention patients and 45…
T_2	caffeine in sport . influence of endurance exercise on the urinary caffeine concentration .
T_1	supervised fitness walking in patients with osteoarthritis of the knee . a randomized , controlled trial .
T_0	the effect of training intensity on ratings of perceived exertion .

Table 2: The top rows contain interleaving of 3 articles with 2 to 5 sentences, bottom contains their interleaved titles.

BASELINES

We reimplemented the Bahdanau et al. (2014) seq2seq model, and evaluated it on a standard task of news summarization. We use a popular, CNN/DailyMail, dataset Napoles et al. (2012), with ≈300k training examples for the purpose. We fetched the test set from See et al. (2017) and report the results on it. The results are compared with the seq2seq methods of Nallapati et al. Nallapati et al. (2016) and See et al. See et al. (2017). As our aim for this experiment is to demonstrate the strength of the reimplementation, we set the parameters to produce comparable results in less computation time (2 days compared to others weeks). We achieve Rouge-1 and Rouge-2 of 29.58 and 9.36 respectively. Nallapati et al. reported 32.49 and 11.84 and See et. al. reported 31.33 and 11.81 for the same. This indicates that our implementation is competitive to the latter tuned models.

We then take the implemented seq2seq model and train and test it on the Easy corpus. We also ran Shang et al. (2018)'s two-step system on the test set of the Easy Interleaved Pubmed corpus. As the Shang et al. (2018) system is unsupervised, it doesn't need training. Additionally, we also utilized Shang et al. (2018)'s clustering component to first cluster the interleaved texts of the corpus, and then the disentangled corpus is used to train the seq2seq model. We refer to the latter as cluster→seq2seq. The performance comparison of Shang et al. (2018) and the two seq2seq models are shown in Table 3. Clearly, seq2seq performs better than Shang et al. (2018), the reason being a seq2seq model trained on a sufficiently large dataset is better at summarization than the unsupervised sentence compression (extractive) method. The lower performance of cluster→seq2seq in comparison to seq2seq shows not only that a disentanglement component is unnecessary but also illustrates the error propagation of disentanglement to summarization.

Model	Rouge-1	Rouge-2	Rouge-L
Shang et al. (2018)	30.37	10.77	20.04
seq2seq	**44.38**	**19.47**	**35.20**
cluster→seq2seq	42.93	18.76	30.68

Table 3: Rouge F1-Scores for seq2seq models on the Pubmed Easy Corpus.

SEQ2SEQ VS. HIER2HIER MODELS

We then compare the proposed hierarchical approach against the seq-to-seq approach in summarizing the interleaved texts by experimenting on the Medium and Hard corpora.

Table 4 shows the experimental results on several corpora; clearly, an increase in the complexity of interleaving scantly impacts the performances. Though, in case of Pubmed, seq2seq depreciates by 1-2 points while hier2hier remains nearly same with increase of complexity. In Stack Exchange corpora, as Hard corpora has more training data (+30k) seq2seq improves slightly. However, a noticeable improvement is observed on changing the decoder to hierarchical, i.e., 1.5-3 Rouge points in Pubmed and 2-4.5 points in Stack Exchange depending on the interleaving complexity.

To understand the impact of hierarchy on the hier2hier model, we perform an ablation study and use the Hard Pubmed corpus for the experiments, and Table 5 shows the results. Clearly, adding hierarchical decoding already provides a boost in the performance. Hierarchical encoding also adds

Corpus Difficulty	Model	PubmedRCT			Stack Exchange		
		Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L
Medium	seq2seq	30.67	11.71	23.80	18.78	03.52	14.73
	hier2hier	**32.78**	**12.36**	**25.33**	**24.34**	**05.07**	**18.63**
Hard	seq2seq	29.07	10.96	21.76	20.21	04.03	14.93
	hier2hier	**33.36**	**12.69**	**24.72**	**24.96**	**05.56**	**17.95**

Table 4: Rouge Recall-Scores of models on the Medium and Hard Corpus.

some improvements to the performance; however, the enhancement attained in training and inference speed by the hierarchical encoding is much more valuable. We will discuss it in depth later.

	Hard Pubmed Corpus		
Model	Rouge-1	Rouge-2	Rouge-L
seq2seq	29.07	10.96	21.76
seq2hier	32.92	11.87	24.43
hier2seq	31.86	11.9	23.57
hier2hier	**33.36**	**12.69**	**24.72**

Table 5: Rouge Recall-Scores of models on the Hard Pubmed Corpus.

Further, we also compiled corpora to mimic real world conversation interleaving, wherein the sequence of summaries for interleaved posts may not follow the sequential occurrence of posts, see 17-18 in Algorithm. 1). An example, a post mention an action in the very beginning of conversation, but does only elaborates it at the end. Therefore, summary corresponding to the action should be at the end. So, we again create Medium and Hard real world mimicking corpora of the Stack Exchange dataset as it is more similar to real world conversations, and perform the same set of experiments. As seen in Table 6, the results show that both the seq2seq and hier2hier models performance are slightly lower as it is tougher task; see Table 4. In addition, the hier2hier model is still consistently better than the seq2seq model.

	Medium Corpus			Hard Corpus		
Model	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L
seq2seq	19.67	03.88	15.37	19.62	03.71	14.90
hier2hier	**23.97**	**05.63**	**18.75**	**24.14**	**05.00**	**17.25**

Table 6: Rouge Recall-Scores of models on the Stack Exchange Medium and Hard Corpus.

Importantly, hier2hier converges much earlier than the seq2seq and also reaches a lower training loss (Figure 5 in Appendix). hier2hier takes $\approx$1.5 days for training on a Tesla V100 GPU, while seq2seq takes $\approx$4.5 days. Therefore, the hier2hier model not only achieves greater accuracy but also reduces training and inference time.

HIERARCHICAL ATTENTION

The contribution of the post-level and high-level attentions in a hierarchical decoder is two-fold: computing the thread representation and re scaling the word-level attentions. To understand the impact of hierarchical attention on the hier2hier model, we perform an ablation study of post-level attentions (γ) and high-level attentions ($\boldsymbol{\beta}$), using the Hard corpus for the experiments.

Table 7 shows the performance comparison. Clearly, γ attention improves the performance (0.5-1) to hierarchical decoding but not a lot. The high-level attention, i.e., $\boldsymbol{\beta}$ is very important as without it the model performances is noticeably reduced (Rouge values decrease from 2-3). Additionally, we also include Li, Luong and Jurafsky, 2015 post-level attention technique in the comparison, as it is the closest hierarchical attention to ours (see Table 1). Though, (Li, Luong and Jurafsky) utilize a softmax obtained $\boldsymbol{\beta}$ to compute thread representation, and in contrast to us, they do not reuse $\boldsymbol{\beta}$ for re-scaling $\boldsymbol{\alpha}$, and thereby, closest to hier2hier($+\gamma - \boldsymbol{\beta}$). Evidently, re-utilization gives hier2hier($+\gamma - \boldsymbol{\beta}$) an edge over (Li, Luong and Jurafsky, 2015). Lastly, removing both the γ and $\boldsymbol{\beta}$)

Model	Rouge-1	Rouge-2	Rouge-L
hier2hier($+\gamma+\beta$)	**33.36**	**12.69**	**24.72**
hier2hier($-\gamma+\beta$)	32.65	12.21	24.23
hier2hier($+\gamma-\beta$)	31.28	10.20	23.49
hier2hier(Li,Luong&Jurafsky)	29.83	09.80	22.17
hier2hier($-\gamma-\beta$)	30.58	10.00	22.96
seq2seq	29.07	10.96	21.76

Table 7: Rouge Recall-Scores of models on the Hard Pubmed Corpus.

makes the hier2hier similar to seq2seq, except a few more parameters, i.e., two additional LSTM, and the performance is also very similar.

	Interleaved Texts
0	this study was conducted **to evaluate the influence of** excessive **sweating** during **long-distance** running **on** the urinary concentration of **caffeine**. . .
1	**to assess the effect of** a **program of** supervised **fitness** walking and patient education on functional status , pain , and. . .
. . .	. . .
5	a total of 102 patients with a documented diagnosis of primary osteoarthritis of one or both knees participated. . .
6	we **examined the effects of intensity of training** on ratings of perceived exertion **(**. . .
. . .	. . .
	GroundTruth/Generation
0, 2	caffeine in sport . influence of endurance exercise on the urinary caffeine concentration . **effect of excessive [UNK] during [UNK] running on the urinary concentration of caffeine .**
1, 4	supervised fitness walking in patients with osteoarthritis of the knee . a randomized , controlled trial . **effect of a physical fitness walking on functional status , pain , and pain**
6, 8	the effect of training intensity on ratings of perceived exertion . **effects of intensity of training on perceived [UNK] in [UNK] athletes .**

Table 8: Interleaved sentences of 3 articles, and corresponding ground-truth and hier2hier generated summaries. The top 2 sentences that were attended (γ) for the generation are in left. Additionally, top words (β) attended for the generation are colored accordingly.

Discussion

Occurrence of interleaved texts is common; however, readers often don't have the patience to wade through them. Surveys have shown that summarized contents are easy to consume, and thus, many organizations have begun displaying extractive summaries, content visualization or the combination alongside the interleaved texts.[1] Table 8 shows an output of our hierarchical abstractive system, in which interleaved texts are in the top, and ground-truth and generated summaries in the bottom. Table 8 also shows the top two post indexes attended to by the post-level attention (γ) while generating those summaries, and they coincide with relevant posts. Similarly, the top 10 indexes (words) of the high-level attention (β) is directly visualized in the table through the color coding matching the generation. The system not only manages to disentangle the interleaved texts but also to generate appropriate abstractive summaries. Meanwhile, β provides explainability of the output.

As interleaving in the table includes abstracts from the same category, e.g. Physical Activities, the interleaving is complex and approximates the real-world conversations. Despite that, the end-to-end hierarchical system tackles the task to a large extent. The next, future stage in this research is transfer learning of the hierarchical system to more types of conversations. We find a Stack-Exchange Hard corpora (a=6, b=10, m=2 and n=3) trained along with the popular meeting AMI corpus (McCowan et al. (2005)) already reaches competitive results (Rouge-1=25.9, Rouge-2=03.10) against SOA abstractive system on AMI (e.g. Rouge-1=21.7 and Rouge-2=02.5 of TextRank (Mihalcea & Tarau (2004)), Rouge-1=27.9, Rouge-2=04.10 of FUSION (Mehdad et al. (2013))). Additionally, we aim to modify hier2hier to include some of the recent additions of seq2seq models, e.g., see et al 2017 copy mechanism.

[1] http://resources.trustyou.com/c/wp-present-travel-review-content?x=0MFT5U

Conclusion

We presented an end-to-end trainable hierarchical encoder-decoder architecture which implicitly disentangles interleaved texts and generates a multi-sentence abstractive summary covering the text threads. Furthermore, the architecture addresses the error propagation issue that occurs in the two-step architectures. Our proposed novel hierarchical attention further boosts both disentanglement and summary generation.

References

Ahmet Aker, Monica Paramita, Emina Kurtic, Adam Funk, Emma Barker, Mark Hepple, and Rob Gaizauskas. Automatic label generation for news comment clusters. In *Proceedings of the 9th International Natural Language Generation Conference*, pp. 61–69, 2016.

Xavier Anguera, Simon Bozonnet, Nicholas Evans, Corinne Fredouille, Gerald Friedland, and Oriol Vinyals. Speaker diarization: A review of recent research. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(2):356–370, 2012.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL `http://arxiv.org/abs/1409.0473`.

Emma Barker, Monica Lestari Paramita, Ahmet Aker, Emina Kurtic, Mark Hepple, and Robert Gaizauskas. The sensei annotated corpus: Human summaries of reader comment conversations in on-line news. In *Proceedings of the 17th annual meeting of the special interest group on discourse and dialogue*, pp. 42–52, 2016.

Sumit Chopra, Michael Auli, and Alexander M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In Kevin Knight, Ani Nenkova, and Owen Rambow (eds.), *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pp. 93–98. The Association for Computational Linguistics, 2016. URL `http://aclweb.org/anthology/N/N16/N16-1012.pdf`.

Franck Dernoncourt and Ji Young Lee. Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 308–313. Asian Federation of Natural Language Processing, 2017. URL `http://aclweb.org/anthology/I17-2052`.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 132–141. Association for Computational Linguistics, 2018. URL `http://aclweb.org/anthology/P18-1013`.

Jyun-Yu Jiang, Francine Chen, Yan-Ying Chen, and Wei Wang. Learning to disentangle interleaved conversational threads with a siamese hierarchical network and similarity ranking. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1812–1822. Association for Computational Linguistics, 2018. doi: 10.18653/v1/N18-1164. URL `http://aclweb.org/anthology/N18-1164`.

Baoyu Jing, Pengtao Xie, and Eric Xing. On the automatic generation of medical imaging reports. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2577–2586. Association for Computational Linguistics, 2018. URL `http://aclweb.org/anthology/P18-1240`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Jonathan Krause, Justin Johnson, Ranjay Krishna, and Li Fei-Fei. A hierarchical approach for generating descriptive image paragraphs. In *Computer Vision and Patterm Recognition (CVPR)*, 2017.

Jiwei Li, Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1106–1115. Association for Computational Linguistics, 2015. doi: 10.3115/v1/P15-1107. URL `http://aclweb.org/anthology/P15-1107`.

Zongyang Ma, Aixin Sun, Quan Yuan, and Gao Cong. Topic-driven reader comments summarization. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pp. 265–274. ACM, 2012.

I. McCowan, J. Carletta, W. Kraaij, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, M. Kronenthal, G. Lathoud, M. Lincoln, A. Lisowska, W. Post, Dennis Reidsma, and P. Wellner. The ami meeting corpus. In L.P.J.J. Noldus, F Grieco, L.W.S. Loijens, and P.H. Zimmerman (eds.), *Proceedings of Measuring Behavior 2005, 5th International Conference on Methods and Techniques in Behavioral Research*, pp. 137–140. Noldus Information Technology, 8 2005. ISBN 90-74821-71-5.

Yashar Mehdad, Giuseppe Carenini, Frank Tompa, et al. Abstractive meeting summarization with entailment and fusion. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pp. 136–146, 2013.

Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pp. 404–411, 2004.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In Yoav Goldberg and Stefan Riezler (eds.), *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pp. 280–290. ACL, 2016. URL `http://aclweb.org/anthology/K/K16/K16-1028.pdf`.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI Conference on Artificial Intelligence*, 2017. URL `https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14636/14080`.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pp. 95–100. Association for Computational Linguistics, 2012.

Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton (eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pp. 379–389. The Association for Computational Linguistics, 2015. URL `http://aclweb.org/anthology/D/D15/D15-1044.pdf`.

Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1073–1083. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1099. URL `http://www.aclweb.org/anthology/P17-1099`.

Guokan Shang, Wensi Ding, Zekun Zhang, Antoine Tixier, Polykarpos Meladianos, Michalis Vazirgiannis, and Jean-Pierre Lorré. Unsupervised abstractive meeting summarization with multi-sentence compression and budgeted submodular maximization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 664–674. Association for Computational Linguistics, 2018. URL `http://aclweb.org/anthology/P18-1062`.

Suzan Verberne, Emiel Krahmer, Iris Hendrickx, Sander Wubben, and Antal van Den Bosch. Creating a reference data set for the summarization of discussion forum threads. *Language Resources and Evaluation*, pp. 1–23, 2018.

Lidan Wang and Douglas W Oard. Context-based message expansion for disentanglement of interleaved text conversations. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*, pp. 200–208. Association for Computational Linguistics, 2009.

Appendix

In Algorithm. 1, INTERLEAVE takes a set of concatenated abstracts and titles, $C = \langle A_1; T_1, \ldots, A_{|C|}; T_{|C|} \rangle$, minimum, a, and maximum, b, number of abstracts to interleave, and minimum, m, and maximum, n, number of sentences in a source, and then returns a set of concatenated interleaved texts and summaries. WINDOW takes a sequence of texts, X, and returns a window iterator of size $\frac{|X|-w}{t} + 1$, where w and t are window size and sliding step respectively. *window* reuses elements of X, and therefore, enlarges the corpus size. Notations $\mathcal{U}$ refers to a uniform sampling, $[\cdot]$ to array indexing, and REVERSE to reversing an array.

Algorithm 1 Interleaving Algorithm

```
1: procedure INTERLEAVE($C, a, b, m, n$)
2:     $\hat{C}, Z \leftarrow$ WINDOW($C, w = b, t = 1$), Array()
3:     while $\hat{C} \neq \emptyset$ do
4:         $C', A', T', S \leftarrow \hat{C}$.NEXT(), Array(), Array(), {}
5:         $r \sim \mathcal{U}(a, b)$
6:         for $j$ = 1 to $r$ do                                    ▷ Selection
7:             $A, T \leftarrow \hat{C}[j]$
8:             $T'$.ADD($T$)
9:             $q \sim \mathcal{U}(m, n)$
10:            $A'$.ADD($A[1{:}q]$)
11:            $S \leftarrow S \cup \{j_{\times q}\}$
12:        $\hat{A}, \hat{T} \leftarrow$ Array(), Array()
13:        for 1 to $|S|$ do                                       ▷ Interleaving
14:            $k \leftarrow \mathcal{U}(S)$
15:            $S \leftarrow S \backslash k$
16:            $I \leftarrow$ REVERSE($A'[k]$).POP()
17:            $\hat{A}$.ADD($I$)
18:            $J \leftarrow T'[k]$
19:            if $J \notin \hat{T}$ then:
20:                $\hat{T}$.ADD($J$)
21:        $Z$.ADD($\hat{A}$;$\hat{T}$)
22:    return $Z$
```

Parameters

For the word-to-word encoder, the steps are limited to 20, while the steps in the word-to-word decoder are limited to 14. The steps in the post-to-post encoder and thread-to-thread decoder depend on the corpus type, e.g., Medium has 15 steps in post-to-post and 3 steps in thread-to-thread. In seq2seq experiments, the source is flattened, and therefore, the number of steps in the source is limited to 300. We initialized all weights, including word embeddings, with a random normal distribution with mean 0 and standard deviation 0.1. The embedding vectors are of dimension 80. The vocabulary size is limited to 5000 and 15000 for Pubmed and Stack Exchange corpora respectively. All hidden states of the encoder and decoder in the models are set to dimension 100. We pad short sequences with a special token, $\langle PAD \rangle$. We use Adam Kingma & Ba (2014) with an initial learning rate of .0001 and batch size of 64 for training. Texts are lowercased and numbers are replaced by the special symbol $\%$.

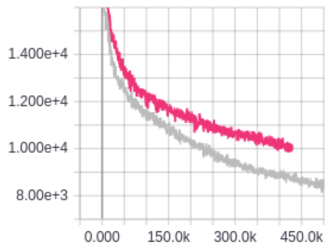

Figure 5: A comparison of running average training loss between seq2seq (pink) and hier2hier (gray) for Stack Exchange Hard corpus.